Introduction

This assignment asks you to construct simple graphical user interface (GUI) applications.

Please make sure you properly understand and **complete part(a) first**.

(b) Bursting for a challenge?

***Note**: Some concepts required to do part (b) may not been explicitly covered in lectures yet. These include using Swing's Timer class and calling repaint() to update the JPanel's display. You should be able to look at the Java API, the examples in the Java Tutorial, and the course textbook, to get the majority of it working without too much trouble.*

The **Balloons** game challenges players to burst as many balloons as they can in the time allowed. It is based on the *Shape* hierarchy you created in Lab03. You may be able to download the jar file and play the basic game (assuming you have the right version of Java installed on your computer, the planets are appropriately aligned, etc., etc.), but if not, don't worry, the real fun is in programming it yourself ☺

From Lab03, you should already have a *Shape* class that implements *Locatable*, a *Circle* class (which is a Shape) that implements *Selectable*, and a *ShapeContainer* class that holds a collection of such selectable shape objects.

Before you start, create a new project folder for Lab04b, and copy your *Shape* hierarchy into it, making sure to put it in a package called *shapes*. To the package, add an interface *Drawable,* which has a single method, "*void draw(Graphics g);*". Make your *ShapeContainer* implement *Iterable* and add two new methods to it: *int size()* ~that returns the number of shapes in the container~ and *int selectAllAt(int x, int y)*. ~that returns the number of shapes containing the point *x,y* and sets the *selected* property of those shapes to true. *Note:* your class should already have a method *void removeSelected()* ~that removes all selected shapes from the container~ and *Shape contains(int x, int y)* ~that returns the first shape that includes the point x, y, or null if none do~, however, it ***should not*** have a method, such as *Shape get(int index)*, that returns the Shape at a specified index location in the collection.

From now on everything should be done in another package (possibly -though preferably not- the default one).

Create a normal application, called *Lab04b*, with a main method. Have it create a *JFrame* into which you can place the balloons game panel. Create a *BalloonsGamePanel* class and add an instance of it to your *JFrame*. In the *BalloonsGamePanel* class, have a *ShapeContainer* property, called *balloons*. Override the *paintComponent* method of the *BalloonsGamePanel* so that it calls the *draw* method of each shape in the container.

Next create a *Balloon* class that extends your *Circle* class and implements *Drawable*. Balloons should have a default radius of 25. Try adding a few balloons (instances of the *Balloon* class), to your *balloons ShapeContainer* to ensure everything is in order! Add a *grow* method to your *Balloon* class, that increases the radius of the balloon (Circle) by a fixed number of units, up to a maximum size, say 100, at which point the circle should be *selected* and the *radius* set to zero.

Add a *javax.swing.Timer* object to your *BalloonsGamePanel*, and have it call the *grow()* method of each *Balloon* in *balloons*, every 250 milliseconds or so. Have the game constructor create say 25 balloons at random locations on the panel.

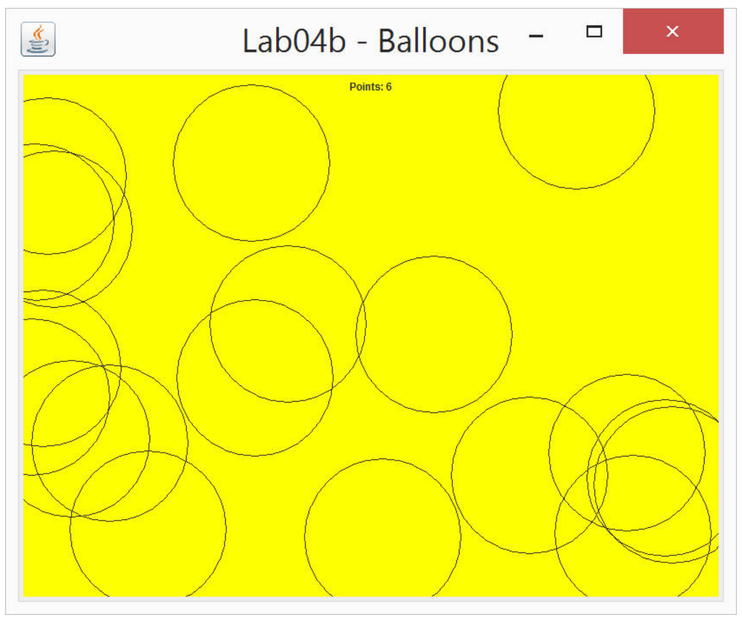
To simulate bursting a balloon with a pin, add a *MouseListener* to your *BalloonsGamePanel* and override the *mousePressed* method so that it calls the *selectAllAt(x, y)* method of *balloons* using the location of mouse pressed event. You will need to call *removeSelected()* to actually remove the balloons from the screen/collection and while this could be done in the mouse listener, it may be more conveniently placed in the timer object's listener.

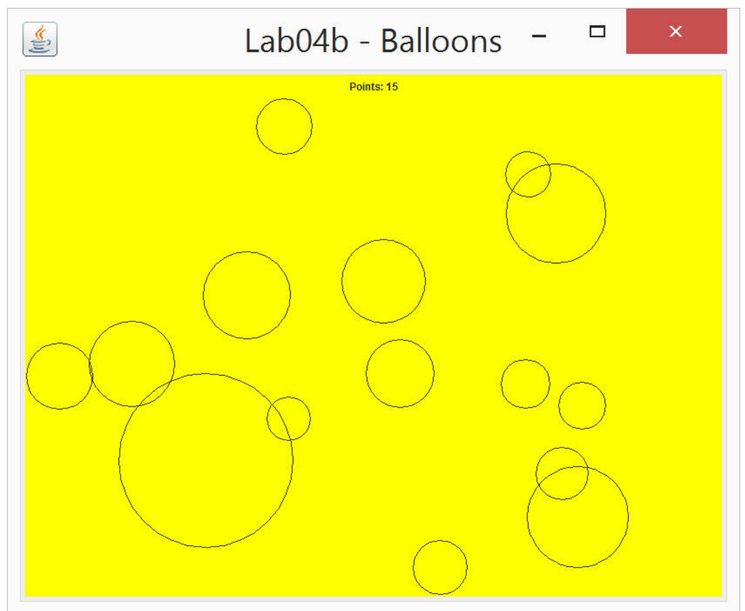
Once all this is working, you simply need to add a few bells 'n whistles to make it into a reasonable game.

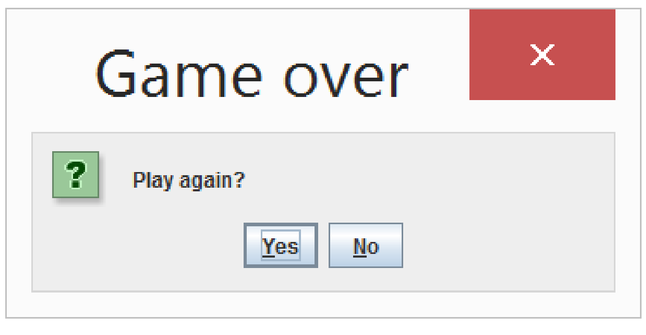
Add a *points* property to your class. Add points only if more than two balloons are burst with a single mouse press. Whenever there are less than say 15 balloons left, add another balloon at a random location ~do this in the timer's listener. Add a *JLabel* to the game panel and use it to display the points scored so far.

Add yet another property, *elapsedTime*, to keep track of how long the game has been running. Increment it in the timer's listener method, and then stop the game after a fixed period. Use a *JOptionPane* to inform the user the game is over. Perhaps allow the user to "Play again?" if they wish.

There are lots of ways you might improve this basic game... use your imagination and show us what you can do.







Have fun!
